异构资源环境下 Hadoop 节点能力自适应调度算法 *

沈学利 [a]，盛方严 [b]

(辽宁工程技术大学 a. 电子与信息工程学院; b. 研究生院，辽宁 葫芦岛 125105)

摘　要：为了解决当前 Hadoop 集群在异构资源环境下固有的调度分配方法的不足，提出了一种基于节点能力的自适应调度算法 NCAS（node capacity adaptive scheduling）。首先，NCAS 算法根据节点性能、任务特征计算得到调度因子；然后，由调度因子确定各节点应分得的数据量与任务槽数；最后，将数据和任务多分给快节点同时少分给慢节点。实验结果表明，与传统的调度算法相比，NCAS 算法大幅度减少了备份任务的启动数量，明显减少了作业完成时间，提升了任务执行效率。

关键词：Hadoop；异构资源；节点能力；自适应

中图分类号：TP301.6　　doi: 10.19734/j.issn.1001-3695.2018.08.0554

Hadoop node capability adaptive scheduling algorithm in Heterogeneous resource environments

Shen Xueli[a], Sheng Fangyan[b]

(*a. School of Electronics & Information Engineering, b. School of Graduate Studies, Liaoning Technical University, Huludao Liaoning* 125105, *China*)

Abstract: In order to solve the shortcomings of current Hadoop cluster inherent scheduling distributed methods in heterogeneous resource environments, this paper proposed an adaptive scheduling algorithm NCAS (Node Capacity Adaptive Scheduling) based on the node capability. Firstly, NCAS algorithm calculated the scheduling factor based on node performance and task characteristics; Secondly, the scheduling factor determined the amount of data and the number of task slot that each node should be assigned; Finally, NCAS algorithm dispatched data and tasks more into fast nodes and less into slow nodes. Experimental results show that, compared with the traditional scheduling algorithm, NCAS algorithm can greatly reduce the number of speculative tasks, significantly reduce the job completion time. It also can improve the task execution efficiency .

Key words: Hadoop; Heterogeneous resource; node capability; adaptive

0　引言

Hadoop[1]作为一个云存储和云计算的开源解决方案，在学术界和工业上都受到越来越广泛地运用。作业调度算法和任务调度算法在 Hadoop 的性能中起着十分重要的作用[2,3]。目前 Hadoop 几种常见的作业调度算法有先进先出调度算法 FIFO(first in first out)[4]、公平调度算法(fair scheduler)[5]、计算能力调度算法(capacity scheduler)[6]等。这些 Hadoop 作业调度算法都是针对同构集群资源的情况设计，在异构资源环境下效率低下[7]。为了解决异构环境下 Hadoop 默认推测式任务调度算法(Speculative Task)[8]的不足，Zaharia 等人[9]提出了 LATE 调度器，通过为最长剩余完成时间的任务启动备份任务提升了 Hadoop 的任务执行效率，但其中估算任务剩余完成时间有待改进；陶永才等人[10]提出了一种自适应 MapReduce 调度器，此算法解决了 Hadoop 在异构资源环境下某种程度上负载分布不均的问题，但其中任务预期完成时间较难确定；郑晓薇等人[11]提出了一种基于节点能力的任务自适应调度方法，它充分考虑了集群中节点资源的异构性，但其易引起节点间数据频繁传输，占用网络带宽，降低系统性能。

为解决上述研究中 Hadoop 调度算法在异构环境下仍存在的运行效率低下问题，提出了节点能力自适应调度算法 NCAS。该算法通过计算得到代表各节点计算能力大小的调度因子的值，根据调度因子为快节点或慢节点分配任务槽数和数据量的多少，达到提高系统整体执行效率的目的。

1　研究背景

Hadoop 的核心主要由两部分构成：分布式文件系统 HDFS(Hadoop distributed file system)和 MapReduce 的实现系统。

Hadoop 集群的系统架构由一个主节点和一组从节点构成。如图 1 所示，HDFS 中主节点和从节点上分别运行一个 NameNode 和 DataNode 进程。NameNode 存储 HDFS 中的元数据，负责响应用户对 HDFS 文件系统的访问；DataNode 负责存储数据，执行 NameNode 的命令，对数据块进行创建、复制和删除，客户端可直接对 DataNode 执行读写操作[12]。HDFS 收到文件后将其分割成大小相等的数据块[13]，并复制三次存储于不同节点。

图 2 所示为 MapReduce 工作框架，用户提交的作业被分成若干 map 任务和 reduce 任务[14]，每个 map 任务处理一个数据块，输出相应的键值对，reduce 任务处理这些键值对得到最终输出结果[15]。每个从节点上有一定数量的 map 任务槽和 reduce 任务槽，一个槽执行一个相应类型的任务，槽配置在 Hadoop 的性能上起着至关重要的作用[16]。为了管理作业

收稿日期：2018-08-21；**修回日期：**2018-10-08　**基金项目：**国家自然科学基金资助项目（61602227）

作者简介：沈学利（1969-），男，江苏连云港人，教授，硕士，主要研究方向为计算机网络、网络安全（523419858@qq.com）；盛方严（1992-），男，硕士研究生，主要研究方向为计算机网络．

的运行，主节点与从节点上分别运行一个 JobTracker 和 TaskTracker 进程。JobTracker 负责调度各个 Map 任务和 Reduce 任务到相应的 TaskTracker 上，并进行任务执行监控与容错管理；TaskTracker 负责执行分得的 Map 任务和 Reduce 任务，收集和存储所得结果，并通过周期性心跳将节点的当前状态与资源利用情况汇报给 JobTracker。系统为落后任务启动备份任务，重新调度执行运行失败的任务。

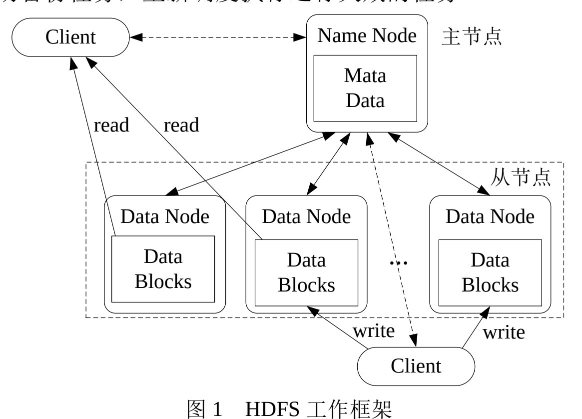


图 1　HDFS 工作框架

Fig. 1　The work framework of HDFS

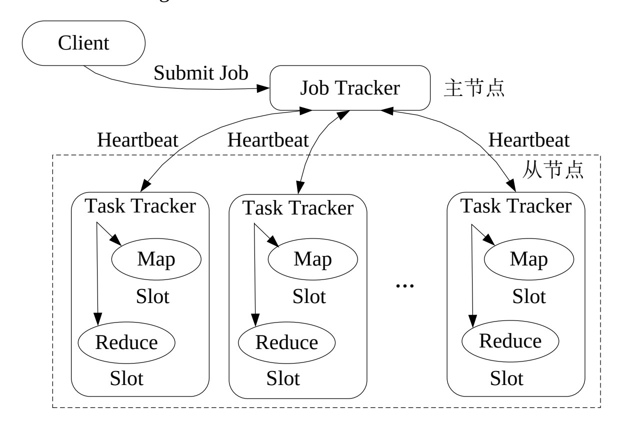


图 2　MapReduce 工作框架

Fig. 2　The work framework of mapreduce

2　NCAS 调度算法

NCAS 算法收到用户的文件后首先对其分割，之后将得到的数据块依次分配到 HDFS 上保存，然后进行 Map 任务调度，以及 Reduce 任务调度。为讲解清楚，首先讲述两个相关概念，然后提出自适应数据分发算法和自适应任务调度算法构成了 NCAS 算法。

2.1　相关概念

2.1.1 调度因子

调度因子与节点性能、任务特征两个因素有关，它代表了节点计算能力的强弱。其中节点性能可由节点的硬件参数确定，硬件参数越大，它的节点性能就越好。在异构环境中，相同槽处理不同作业时表现出不同的运算速度[17]，定义任务特征来表示同一节点执行不同任务时表现出的不同性能。Hadoop 集群处理的作业类型主要有三种，分别为 CPU 密集型（CPU-bound）、I/O 密集型（I/O-bound）和二者的混合型（class sway）[18]。本文中选用 CPU-bound 型和 I/O-bound 型的任务，以节点从前执行不同类型任务所花时间为参数计算得到任务特征的值。随着任务不断执行，任务特征参数可能发生变化，设置一定周期更新计算任务特征参数。本文通过脚本读取并计算各种节点信息值，把结果保存在文件中。

针对资源集 R={r~1~,r~2~,⋯,r~m~}，用 N_{rj} 和 T_{tj} 分别表示节点 r_j(1≤j≤m)的节点性能和任务特征，W_j 为其调度因子。节点 r_j 的 CPU 主频、内存容量、网络带宽和磁盘最大读写速率分别为 tt_{cj}、tt_{mj}、tt_{nj} 和 tt_{dj}，集群平均的 CPU 主频、内存容量、网络带宽和磁盘最大读写速率分别为 avg_c、avg_m、avg_n 和 avg_d。节点性能的计算为：

$$N_{rj} = w_1 * \frac{tt_{cj}}{avg_c} + w_2 * \frac{tt_{mj}}{avg_m} + w_3 * \frac{tt_{nj}}{avg_n} + w_4 * \frac{tt_{dj}}{avg_d} \tag{1}$$

$$\sum_{i=1}^{4} w_i = 1 \tag{2}$$

其中：w_1、w_2、w_3 和 w_4 分别为 CPU 主频、内存容量、网络带宽和磁盘最大读写速率在节点性能中的权重值，权重值之和满足式(2)。

由本地用户日志可查到节点完成任务的历史时间记录，计算得到节点 r_j 运行 Map 任务 CPU-Bound 型与 I/O-Bound 型的平均完成时间为 tt_{cmj} 和 tt_{iomj}，运行 Reduce 任务 CPU-Bound 型与 I/O-Bound 型的平均完成时间为 tt_{crj} 与 tt_{iorj}，集群运行 Map 任务 CPU-Bound 型与 I/O-Bound 型平均完成时间为 avg_{cm} 和 avg_{iom}，运行 Reduce 任务 CPU-Bound 型与 I/O-Bound 型的平均完成时间为 avg_{cr} 和 avg_{ior}。任务特征 T_{tj} 表示为

$$T_{tj} = \begin{cases} \frac{1}{tt_{cmj}/avg_{cm}} + \frac{1}{tt_{iomj}/avg_{iom}}, \text{Map 阶段;} \\ \frac{1}{tt_{crj}/avg_{cr}} + \frac{1}{tt_{iorj}/avg_{ior}}, \text{Reduce 阶段.} \end{cases} \tag{3}$$

调度因子 W_j 的计算式为

$$W_j = w_5 * N_{rj} + w_6 * T_{tj} \tag{4}$$

$$w_5 + w_6 = 1 \tag{5}$$

其中：w_5、w_6 分别为节点性能、任务特征在调度因子中的权重值，权重值之和满足(5)式。

2.1.2 节点异构性

用异构因子 *HF*(heterogeneity factor)表示集群资源的计算能力异构性。定义 *HF* 为集群资源集 $R=\{r_1,r_2,\cdots,r_m\}$ 的平均异构因子，不同作业使用 CPU 和 I/O 资源的程度不同，所以 *HF* 因运行完不同类型任务而不同。*HF* 的表达式为

$$HF = \sqrt{D(X)} = \sqrt{E(X - E(X))^2} \tag{6}$$

x_j 是 r_j 的调度因子，如式(7)所示。E(X)是 $X=\{x_1,x_2,\cdots,x_m\}$

$$\begin{cases} x_1 = W_1 \\ x_2 = W_2 \\ \vdots \\ x_m = W_m \end{cases} \tag{7}$$

$$E(X) = \sum_{j=1}^{m} x_j \cdot P_j \tag{8}$$

的均值，通过式(8)计算。其中节点 r_j 分得任务的几率为 P_j，集群中有 m 个节点，P_j 为 1/m，D(X)是 X 的均方差。

2.2　自适应数据分发算法

在异构资源环境下，各节点计算能力差别大，每个节点分得相同的数据量，很快处理完本节点数据的快节点需处理慢节点上的数据，远距离快慢节点间传输大量数据需巨额花费。为尽量避免产生这种情况[19]，使各节点存储的数据量和它的计算能力大小成正比，需远程传输的数据量将减少很多[20]。自适应数据分发算法根据调度因子将大量数据分配给快节点；少量数据分给慢节点，避免远程传输大量数据，占用

网络带宽。

异构环境下节点 r_j 的数据分配量 $N_D(r_j)$通过下式计算：

$$N_D(r_j)=\frac{N_D}{m}\times\frac{x_j}{E(X)} \tag{9}$$

其中：N_D 为用户作业数据量（此时 x_j 中的 T_{ij} 取 Map 阶段的值）。

自适应 DFS 数据分发算法如下：

a）对资源节点集合 R 中的每一个节点 r_j，通过式(1)～(5)计算 W_j，令 $x_j=W_j$；

b）通过式(8)计算 E(X)，对 R 中的每一个节点 r_j，通过(6)式计算 HF；

c）比较 HF 与设定的异构因子阈值 $HF_{threshold}$ 的大小关系，如果 $HF<HF_{threshold}$，则说明集群资源为同构，各节点均分 N_D；否则集群资源为异构，按式(9)计算，各节点根据所得结果分配数据量。

2.3　自适应任务调度算法

当前的调度方法面对异构集群时会产生负载分布不均的问题。为尽可能在某种程度上均衡负载，提出自适应任务调度算法。它根据调度因子运算，得到与各节点计算能力大小相匹配的任务槽数进行分配。自适应任务调度算法包含两部分，先介绍其中的自适应 Map 任务调度：

异构环境下分配给节点 r_j 的任务槽数 $N_{slot}(r_j)$通过下式计算：

$$N_{slot}(r_j)=\frac{x_j}{E(X)}\times\overline{N_{slot}} \tag{10}$$

其中: $\overline{N_{slot}}$ 为系统预先设定均分的任务槽数（此时 x_j 中的 T_{ij} 取 Map 阶段的值）。

自适应 Map 任务调度算法如下：

第一步

a）对资源节点集合 R 中的每一个节点 r_j，通过式(1)～(5)计算 W_j，令 $x_j=W_j$；

b）通过式(8)计算 E(X)，对 R 中的每一个节点 r_j，通过式(6)计算 HF；

c）比较 HF 与设定的异构因子阈值 $HF_{threshold}$ 的大小关系，如果 $HF<HF_{threshold}$，则说明集群资源为同构，各节点任务槽数设定为 $\overline{N_{slot}}$；否则集群资源为异构，各节点设定式(10)计算所得的任务槽数 $N_{slot}(r_j)$。

第二步

d）若某个 TaskTracker 上存在空闲 map 任务槽，JobTracker 为其分配未被调度的 map 任务；

e）TaskTracker 定期向 JobTracker 发送心跳信号，汇报节点运行状态；

f）当有 map 任务完成时，TaskTracker 向 JobTracker 发送心跳信息，JobTracker 为其设定式(10)计算所得的任务槽数。

自适应 reduce 任务调度与自适应 map 任务调度类似，不同之处在于 x_j 中 T_{ij} 取 reduce 阶段的值，为避免文章冗长，在此不做介绍。

3　验证及分析

为验证算法，搭建一个由三个机架组成的集群。每个机架由一组不同配置的计算机节点组成，机架内部与机架之间都由交换机连到一起，各个计算机节点的参数见表 1。

表 1　实验资源节点配置

Table 1　Experimental resource node configuration

		第 1 组	第 2 组	第 3 组
节点配置	节点数/个	3	2	2
	节点编号	1,2,3	4,5	6,7
	CPU 主频/MHz	3072	2457	3379
	内存容量/GB	2	2	2
	网络带宽/(Mb • s^{-1})	100	100	100
	最大磁盘读写速率/(MB • S^{-1})	分别为 70.02,72.51, 71.36	分别为 80.50, 77.69	分别为 85.23, 84.69
	核心数	2	2	2

实验中选用的测试程序分别为属于 CPU-bound 型任务的 grep-count、属于 I/O-bound 型任务 TeraSort 和属于 class sway 型任务的 WordCount。根据集群的运行情况，设置一定的周期，更新任务特征参数并从文件中获取任务特征的修正值。

集群在推测式任务调度算法下分别运行 grep-count 和 TaraSort 程序 10 次，每个程序的数据量为 3.0GB。记录运行过程中产生的 map 任务与 reduce 任务的运行信息，经过计算可得任务特征初值，各节点平均历史时间数据记录如表 2 所示。

实验从 Hadoop 启动的备份任务数和作业完成时间两个方面来评价传统调度算法和 NCAS 算法的性能。

表 2　各节点的历史运行信息

Table 2　Historical operation information of each node

节点编号	map 任务平均运行时间/s		reduce 任务平均运行时间/s	
	Grep-Count	TeraSort	Grep-Count	TeraSort
1	13	19	70	220
2	14	24	78	285
3	25	52	112	392
4	22	20	109	226
5	28	22	135	263
6	9	15	52	188
7	11	16	58	202
所有节点均值	18	24	88	254

图 3 所示为启动作业数据量分别为 2 GB、4 GB 和 8 GB 的 WordCount 时，现有 Hadoop 中默认的推式任务调度算法与 NCAS 算法启动备份任务数的对比图。由图可知，NCAS 算法与推测式任务调度算法相比，启用的备份任务数平均减少了 70.0%。分析原因如下：现有 Hadoop 默认集群中的资源是同构的，各节点均分任务槽数和数据量，慢节点上很多任务运行落后，使系统启动了许多的备份任务；而 NCAS 算法依据调度因子为各节点确定与其计算能力相匹配的任务槽数和数据量，使集群运行过程中各节点的进度大小相对比较均衡，减少了备份任务的启动数量，集群的性能得到了优化。

实验程序数据量设为 4 GB，将 Hadoop 集群分别运行在未启动推测式任务调度算法的 Hadoop 调度器、推测式任务调度算法、LATE 算法和 NCAS 算法下，每种任务类型的程序在四种算法下各执行 3 次取均值，获得的运行信息如图 4 所示。

由图 4 可知，各类型测试程序下系统采用未启动推测式任务调度算法的 Hadoop 调度器运行时间都要远多于其他三种算法。NCAS 算法与推测式任务调度算法相比，运行时间

平均减少了 32.2%；与 LATE 算法相比，运行时间平均减少了 16.6%。主要原因是推测式任务调度算法中 Hadoop 默认集群在同构资源环境下，慢节点启动了大量的备份任务；LATE 算法对落后任务的判定更加精确，但未从根本上解决推测式任务调度算法遇到的问题；NCAS 算法为每个节点确定与其计算能力大小成正比的任务槽数和数据量，使运行时间明显缩短。

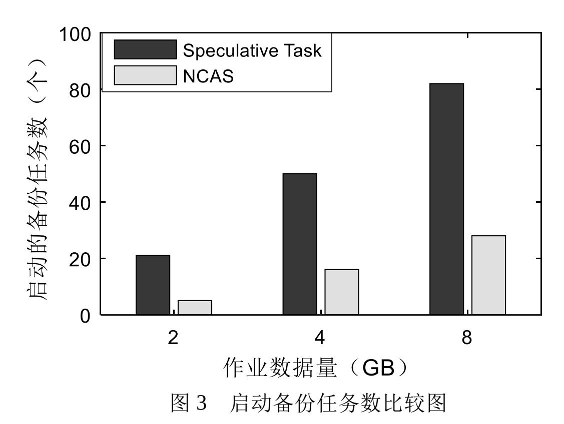


图 3　启动备份任务数比较图

Fig. 3　Comparison of running backup tasks

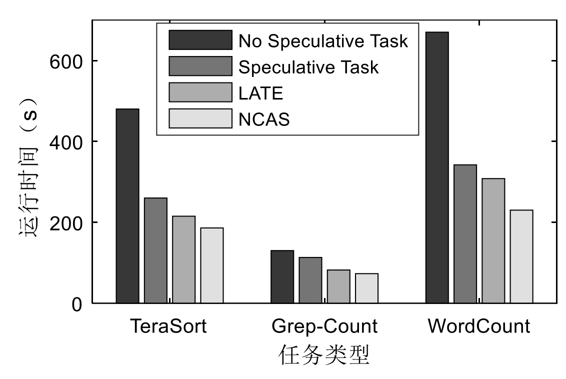


图 4　不同算法运行时间对比图

Fig. 4　Comparison of different algorithm running time

4　结束语

本文研究了 Hadoop 的主要架构及运行原理，提出了异构资源环境下节点能力自适应调度算法 NCAS。该算法通过对节点计算能力相关量的计算得到调度因子，由调度因子所占集群调度因子总和的比例来确定节点应分得的数据量和任务槽数，快节点多分慢节点少分，使集群负载一直处于相对均衡的状态。实验结果表明，NCAS 算法提高了 Hadoop 在异构资源环境下的资源利用率，提升了 Hadoop 的系统性能。

今后的工作中，在节点性能和调度因子中权重值的设定、可能难以估算任务剩余完成时间、可能占用大量网络带宽，降低系统性能方面，对本文算法将做进一步的完善与优化。随着 Hadoop 集群的不断发展，影响节点性能的因素也在不断变化，将对任务调度算法做更加深入的研究。

参考文献：

[1] Apache Hadoop. What is Apache Hadoop [EB/OL]. (2018-06-13) [2018-07-03]. http://hadoop. apache. org/

[2] Han Jiazhen, Yuan Zhengheng, Han Yiheng, *et al.* An adaptive scheduling algorithm for heterogeneous Hadoop systems [C]// Proc of the 16th IEEE/ACIS International Conference on Computer and Information Science. Piscataway, NJ: IEEE Press, 2017: 845-850.

[3] Liu Yyu, Zhang Changjie, Li Bo, *et al.* DeMS: a hybrid scheme of task scheduling and load balancing in computing clusters [J]. Journal of Network and Computer Applications, 2017, 83(1): 213-220.

[4] 王佳琪. Hadoop 平台下调度算法及其改进策略研究 [D]. 北京：北京邮电大学, 2016.

[5] The Apache Software Foundation. Fair Scheduler [EB/OL]. (2013) [2018-07-03]. https://hadoop. apache. org/docs/r1. 2. 1/fair_scheduler. html.

[6] 张聪. Hadoop 分布式系统调度算法的研究 [D]. 北京：北京交通大学, 2016.

[7] 刘莹，罗兴宇，王宁，等. 基于任务进度感知的异构 Hadoop 云平台任务调度方案 [J]. 计算机应用研究, 2017, 34(10): 3139-3143. (Liu Ying, Luo Xingyu, Wang Ning, *et al.* Task scheduling scheme for heterogeneous Hadoop cloud platform based on task progress awareness [J]. Application Research of Computers, 2017, 34(10): 3139-3143.)

[8] 刘奎，刘向东，马宝来，等. 基于数据局部性的推测式 Hadoop 任务调度算法研究 [J]. 计算机应用研究, 2014, 31(1): 182-187. (Liu Kui, Liu Xiangdong, Ma Baolai, *et al.* Research on speculative Hadoop task scheduling algorithm based on data locality [J]. Application Research of Computers, 2014, 31(1): 182-187.)

[9] Zaharia M, Konwinski A, Joseph A D, *et al.* Improving MapReduce performance in heterogeneous environments [C]// Proc of the 8th USENIX Conference on Operating Systems Design and Implementation. New York: ACM Press, 2008: 29-42.

[10] 陶永才，石磊. 异构环境下 MapReduce 性能优化 [J]. 小型微型计算机系统, 2013, 34(2): 287-292. (Tao Yongcai, Shi Lei. Optimization of MapReduce performance in heterogeneous environments [J]. Journal of Chinese Computer Systems, 2013, 34(2): 287-292.)

[11] 郑晓薇，项明，张大为，等. 基于节点能力的 Hadoop 集群任务自适应调度方法 [J]. 计算机研究与发展, 2014, 51(3): 618-626. (Zheng Xiaowei, Xiang Ming, Zhang Dawei, *et al.* Hadoop cluster task adaptive scheduling method based on node capability [J]. Journal of Computer Research and Development, 2014, 51(3): 618-626.)

[12] 陈文龙. Hadoop 平台下作业调度算法研究 [D]. 南京：南京理工大学, 2015. （ChenWenlong. Research on job scheduling algorithms based on Hadoop [D].Nanjing: Nanjing University of Science and Technology,2015.）

[13] Guo Zhenhua, Pierce M, Fox G, *et al.* Automatic task re-organization in MapReduce [C]// Proc of IEEE International Conference on Cluster Computing. Piscataway, NJ: IEEE Press, 2011: 335-343.

[14] Chen Quan, Zhang Daqiang, Guo Minyi, *et al.* SAMR: a self-adaptive MapReduce scheduling algorithm in heterogeneous environment [C]// Proc of the 10th IEEE International Conference on Computer and Information Technology. Piscataway, NJ: IEEE Press, 2010: 2736-2743.

[15] Lim B, Shim Y, Chung Y D. 2PTS: a two-phase task scheduling algorithm for MapReduce [J]. IEICE Trans on Information & Systems, 2016, 99 (9): 2377-2380.

[16] Wang Jiayin, Yao Yi, Mao Ying, *et al.* FRESH: fair and efficient slot configuration and scheduling for Hadoop clusters [C]// Proc of the 7th IEEE International Conference on Cloud Computing. Piscataway, NJ: IEEE Press, 2014: 761-768.

[17] Tang Zhuo, Liu Min, Ammar A, *et al.* An optimized MapReduce workflow scheduling algorithm for heterogeneous computing [J]. Journal of Supercomputing, 2014, 72(6): 1-21.

[18] Tian Chao, Zhou Haojie, He Yongqiang, *et al.* A dynamic MapReduce scheduler for heterogeneous workloads [C]// Proc of the 8th International Conference on Grid and Cooperative Computing,

Piscataway, NJ: IEEE Press, 2009: 218-224.
[19] 林常航, 郭文忠, 陈煌宁. 针对 Hadoop 异构集群节点性能的数据分配策略 [J]. 小型微型计算机系统, 2015, 36(1): 83-88. (Lin Changhang, Guo Wenzhong, Chen Huangning. Data allocation strategy for Hadoop heterogeneous cluster node performance [J]. Journal of Chinese Computer Systems, 2015, 36(1): 83-88.)
[20] 何翔, 李仁发, 唐卓. 一种异构环境下的基于 MapReduce 任务调度改进机制 [J]. 计算机应用研究, 2013, 30(11): 3370-3373. (He Xiang, Li Renfa, Tang Zhuo. An improved mechanism based on MapReduce task scheduling in heterogeneous environments [J]. Application Research of Computers, 2013, 30(11): 3370-3373.)